Введение в язык SQL

Вопросы лекции

1. Язык SQL в СУБД (MS Access): назначение, стандарты, достоинства.
2. Структура команд SQL. Типы данных. Выражения.
3. Функциональные возможности языка SQL.
4. Создание запросов.
   в СУБД Access средствами SQL.
5. Диалекты языка SQL в СУБД.
6. Управление транзакциями.
7. Встраивание SQL в прикладные программы

Язык SQL в СУБД (MS Access): назначение, стандарты, достоинства.

В начале 1970-х годов в одной из исследовательских лабораторий компании IBM была разработана экспериментальная реляционная СУБД IBM System R, для которой затем был создан специальный язык SEQUEL, позволявший относительно просто управлять данными в этой СУБД. Аббревиатура SEQUEL расшифровывалась как Structured English QUEry Language — «структурированный английский язык запросов». Позже по юридическим соображениям язык SEQUEL был переименован в SQL.

Целью разработки было создание простого непроцедурного языка, которым мог бы воспользоваться любой пользователь, даже не имеющий навыков программирования. Собственно разработкой языка запросов занимались Дональд Чэмбэрлин (Donald D. Chamberlin) и Рэй Бойс (Ray Boyce).

Изначально, SQL был основным способом работы пользователя с базой данных и представлял собой небольшую совокупность команд (операторов) допускающих:

- создание таблиц,
- добавление в таблицы новых записей,
- извлечение записей из таблиц (в соответствии с заданным условием),
- удаление записей
- изменение структур таблиц.

В связи с усложнением язык SQL стал более прикладным языком программирования, а пользователи получили возможность использовать визуальные построители запросов.

Рассматриваемый язык SQL ориентирован на операции с **данными**, представленными в виде **логически** взаимосвязанных совокупностей **таблиц-отношений**. Важнейшая особенность его структур – ориентация на конечный результат обработки данных, а не на процедуру этой обработки.

Язык SQL сам определяет, где находятся данные, индексы и даже какие наиболее эффективные последовательности операций следует использовать для получения результата, а потому указывать эти детали в запросе к базе данных не требуется.

SQL (*Structured Query Language* - структурированный язык запросов) – непроцедурный язык взаимодействия приложений и пользователей с реляционными СУБД, состоящий из набора стандартных команд на английском языке.

Отдельные команды изначально никак логически не связаны друг с другом.

Базовый вариант SQL содержит порядка **40** команд (часто еще называемых **запросами** или **операторами**) для выполнения различных действий внутри СУБД.

Интерактивный SQL

Для выполнения запросов. Используется для функционирования непосредственно в базе данных чтобы производить вывод для использования его заказчиком. В этой форме SQL, когда вы введете команду, она сейчас же выполнится и вы сможете увидеть вывод (если он вообще получится) - немедленно.

Вложенный SQL

- Для построения прикладных программ. Состоит из команд SQL помещенных внутри программ, которые обычно написаны на некотором другом языке

Язык SQL- стандарты

Поскольку к началу 1980-х годов существовало несколько вариантов СУБД от разных производителей, причём каждый из них обладал собственной реализацией языка запросов, было принято решение разработать стандарт языка, который будет гарантировать переносимость ПО с одной СУБД на другую (при условии, что они будут поддерживать этот стандарт).

Стандарт на язык SQL был выпущен Американским национальным институтом стандартов (ANSI) **в 1986 г., а в 1987 г.** Международная организация стандартов (ISO) приняла его в качестве международного. Дальнейшее развитие языка поставщиками СУБД потребовало принятия в 1992 году нового расширенного стандарта (ANSI **SQL-92** или просто **SQL2**). Следующим стандартом стал SQL:1999 (SQL3). В настоящее время действует стандарт, принятый в 2003 году (**SQL:2003**) с небольшими модификациями, внесёнными позже.

Достоинства языка SQL:

1. Независимость от конкретных СУБД. Если при создании БД не использовались нестандартные возможности языка SQL предоставляемые некоторой СУБД, то такую БД можно без изменений перенести на СУБД другого производителя. К сожалению большинство БД используют особенности СУБД, на которой работают, что затрудняет их перенос на другую СУБД без изменений;
2. Реляционная основа. Реляционная модель имеет солидный теоретический фундамент. Язык SQL основан на реляционной модели и является единственным языком для реляционных БД;

Достоинства языка SQL:

3. SQL обладает высокоуровневой структурой, напоминающей английский язык.
4. SQL позволяет создавать различные представления данных для различных пользователей;
5. SQL является полноценным языком для работы с БД;
6. Стандарты языка SQL. Официальный стандарт языка SQL опубликован ANSI и ISO в 1989 году и значительно расширен в 1992 году.
7. Декларативность С помощью SQL программист описывает только то, какие данные нужно извлечь или модифицировать. То, каким образом это сделать, решает СУБД непосредственно при обработке SQL-запроса.
8. Поддержка архитектуры клиент/ сервер.

Язык SQL - недостатки

Несоответствие реляционной модели данных Создатели реляционной модели данных Эдгар Кодд Создатели реляционной модели данных Эдгар Кодд, Кристофер Дейт и их сторонники указывают на то, что SQL не является истинно реляционным языком.

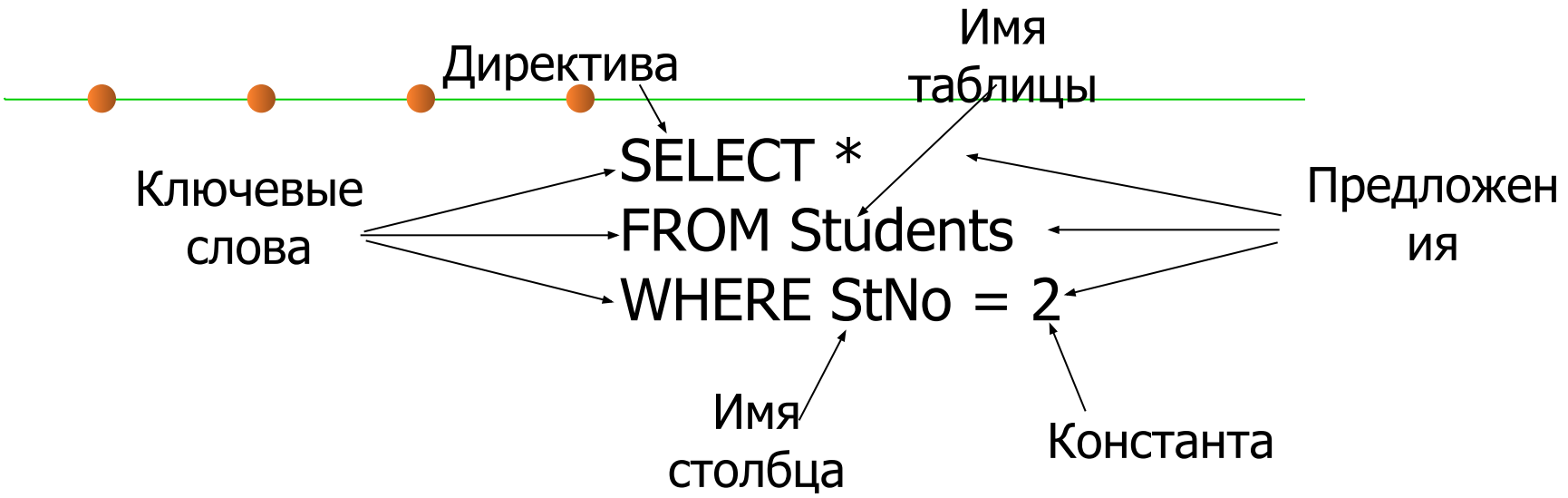
Сложность Хотя SQL и задумывался как средство работы конечного пользователя, в конце концов он стал настолько сложным, что превратился в инструмент программиста.

Отступления от стандартов Несмотря на наличие международного стандарта, многие компании, занимающиеся разработкой СУБД (например, Oracle Несмотря на наличие международного стандарта, многие компании, занимающиеся разработкой СУБД (например, Oracle, Sybase Несмотря на наличие международного стандарта, многие компании, занимающиеся разработкой СУБД (например, Oracle, Sybase, Microsoft Несмотря на наличие международного стандарта, многие компании, занимающиеся разработкой СУБД (например, Oracle, Sybase, Microsoft, MySQL AB), вносят изменения в язык SQL, применяемый в разрабатываемой СУБД. Таким образом, появляются специфичные для каждой конкретной СУБД диалекты языка SQL.

Сложность работы с иерархическими структурами Ранее

Структура команд SQL. Типы данных. Выражения.

1 Структура **команды** SQL



Директивы описывают действие, выполняемое оператором: SELECT (выбрать), CREATE (создать), INSERT (добавить), DELETE (удалить), UPDATE (обновить), DROP (удалить), ALTER (изменить), COMMIT (завершить и зафиксировать внесенные изменения), ROLLBACK (отменить внесенные изменения).

Предложение описывает данные, с которыми работает команда (оператор), или содержит уточняющую информацию о действии, выполняемом оператором: FROM (откуда), WHERE (где), GROUP BY (группировать по), HAVING (имеющий), ORDER BY (упорядочить по), INTO (куда).

2 Имена (идентификаторы)

- длина – до 128 символов
- используемые символы – только прописные или строчные буквы латинского алфавита, цифры или символ подчеркивания (_). *Первым символом* должна быть буква.
- составное имя – идентификатор базы данных, ее владельца и (или) объекта базы данных. Например, *полное имя таблицы* состоит из имени владельца таблицы и имени таблицы, разделенных точкой (.): Admin.Students.

3. Комментарии

- /* и */ – многострочный комментарий
- -- – однострочный комментарий

4. Типы данных

- INTEGER или INT – целое число (обычно до 10 значащих цифр и знак);
- SMALLINT – "короткое целое" (обычно до 5 значащих цифр и знак);
- NUMERIC(p, q) – десятичное число, имеющее p цифр (0<p<16) и знак; с помощью q задается число цифр справа от десятичной точки (q< p, если q = 0, оно может быть опущено);
- REAL – число с плавающей запятой;

- CHAR(n) (CHARACTER(n)) – символьная строка фиксированной длины из n символов (0 < n< 256);
- VARCHAR (n) (CHARACTER VARYING (n))– символьная строка переменной длины, не превышающей n символов (n> 0 и разное в разных СУБД, но не более 8 Кб);
- DATE – дата в формате, определяемом специальной командой (по умолчанию mm/dd/yy, например, 10/03/12).
- TIME – время в формате, определяемом специальной командой, по умолчанию hh.mm.ss.
- BOOLEAN – принимает истинностные значения (TRUE или FALSE).

5. Константы (литералы)

- Числовые константы: 21, -345, +234,6547
  - Константы с плавающей запятой: 1.5E3, -3.14159E1, 2.5 E7
- Строковые константы: ‘Это символьная строка’.

  Если в строковую константу нужно включить одинарную кавычку, то вместо нее надо писать две одинарные кавычки:

  ‘ Здесь внутри будут ‘ ‘ одинарные’ ’ кавычки’.
- Константы даты и времени. Пример для даты: ‘2012-10-03’, ‘1993-12-10’. Пример для времени: ’17:22:10’, ’01:01:01’.
- Логические константы: TRUE, FALSE, UNKNOWN
- Отсутствующие данные (значение NULL)

6 Функции - это операции, позволяющие манипулировать данными. Можно выделить несколько групп встроенных функций:

- Строковые функции. Используются для управления текстовыми строками, например, для обрезания или заполнения значений.
- Числовые функции. Используются для выполнения математических операций над числовыми данными. К числовым функциям относятся функции возвращающие абсолютные значения, синусы и косинусы углов, квадратный корень числа и т.д. Используются они только для алгебраических, тригонометрических и геометрических вычислений.
- Итоговые функции. Используются для получения итоговых данных по таблицам, например, когда надо просуммировать какие-либо данные без их выборки.
- Функции даты и времени. Используются для управления значениями даты и времени, например, для возвращения разницы между датами.
- Системные функции. Возвращают служебную информацию СУБД.

Итоговые функции

- AVG() Функция возвращает среднее значение столбца.
- COUNT() Функция возвращает число строк в столбце.
- MAX() Функция возвращает самое большое значение в столбце.
- MIN() Функция возвращает самое маленькое значение в столбце.
- SUM() Функция возвращает сумму значений столбца.

id_user	name	email	password
2	sasha	sergey@mail.ru	1111
3	masha	maxa@mail.ru	2222
4	pasha	pasha@mail.ru	3333

```
SELECT Sum(users.id_user) AS [Sum]
FROM users ;
```

Sum
9

```
SELECT MAX(users.id_user) AS [max]
FROM users ;
```

max
4

```
SELECT CUNT(users.id_user) AS [count]
FROM users ;
```

count
3

```
SELECT MIN(users.id_user) AS [min]
FROM users ;
```

min
2

```
SELECT AVG(users.id_user) AS [Avg]
FROM users ;
```

Avg
3

Встроенные функции

- CAST (значение AS тип данных) – значение, преобразованное к типу данных (например, дата преобразованная в строку)
- CHAR_LENGTH (строка) – длина строки символов
- CURRENT_DATE – текущая дата
- CURRENT_TIME (точность) – текущее время с указанием точности дробной части секунд
- EXTRACT (часть FROM значение) – указанная часть (YEAR, MONTH, DAY, HOUR, MINUTE, SECOND) из значения временного типа: EXTRACT(YEAR FROM DATE ‘2012-10-03’) = 2012
- LOWER(строка), UPPER(строка) – строка, преобразованная к нижнему (верхнему регистру)

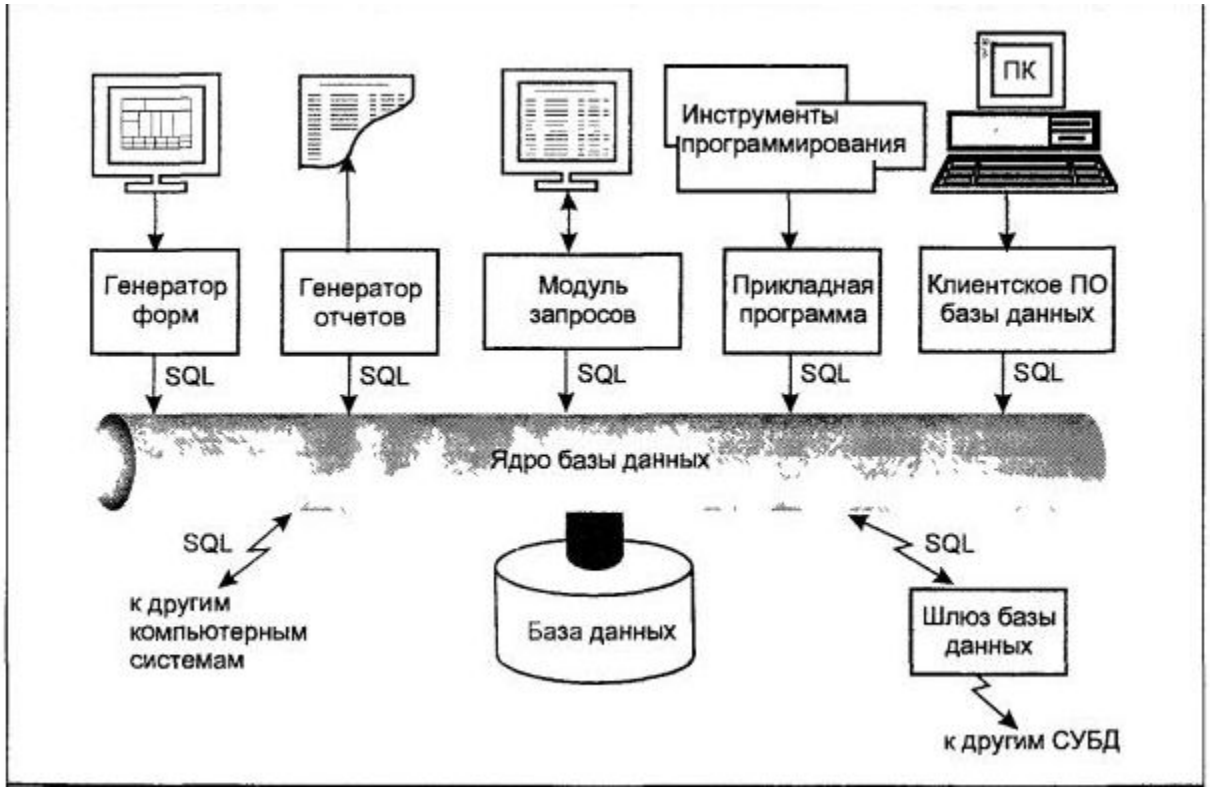
Операторы

- Оператор - это символ, обозначающий действие, выполняемое над одним или несколькими выражениями. Операторы наиболее часто используются в инструкциях DELETE, INSERT, SELECT или UPDATE, а также часто применяются при создании объектов базы данных, таких, как хранимые процедуры, функции, триггеры и представления.

Категории операторов:

- *Арифметические операторы.* Поддерживаются всеми базами данных.
- *Операторы присваивания.* Поддерживаются всеми базами данных.
- *Побитовые операторы.* Поддерживаются Microsoft SQL Server.
- *Операторы сравнения.* Поддерживаются всеми базами данных.
- *Логические операторы.* Поддерживаются в DB2, Oracle, SQL Server и PostgreSQL. *Унарные операторы.* Поддерживаются в DB2, Oracle и SQL Server.

Функциональные возможности языка SQL.



- **SQL – язык интерактивных запросов. Пользователи вводят команды SQL в интерактивном режиме для выборки данных и отображения их на экране, а также внесения изменений в БД.**
- **SQL – язык программирования баз данных. Чтобы получить доступ к базе данных, в прикладные программы вставляются команды SQL.**
- **SQL – язык администрирования баз данных. Администратор БД использует SQL для определения структуры базы данных и управления доступом к данным.**
- **SQL – язык создания приложений клиент/сервер. В прикладных программах SQL используется как средство организации связи по локальной сети с сервером баз данных, в которой хранятся совместно используемые данные.**

Для описания форматов команд языка SQL *используется так называемая форма Бэкуса-Наура* (*БНФ*), *в которой приняты следующие обозначения*:

- < > (*угловые скобки*) – *то, что в них указано, определяет пользователь*;
- [ ] (*квадратные скобки*) – *выделяют те части команды, которые могут отсутствовать*;
- { } (*фигурные скобки*) – *объединяют последовательность элементов в логическую группу*;
- **…** (*многоточие*) – *указывает на допустимость повторения элемента или группы элементов один или несколько раз*;
- ½ (*вертикальная черта*) – *означает альтернативный выбор*;
- ( ) (*круглые скобки*) – *заключают аргументы команды*;
- (*пробелы*) – *используются для разделения элементов команды*.

Типы команд SQL

- **DQL** (Data Query Language – язык запросов) – запросы на извлечение данных из таблиц и описания внешнего вида полученных данных (оператор SELECT).
- **DML** (Data Manipulation Language – язык манипулирования данными) – добавление, удаление и изменение данных (операторы INSERT, DELETE, UPDATE).

SELECT - отобрать строки из таблиц

INSERT - добавить строки в таблицу

UPDATE - изменить строки в таблице

DELETE - удалить строки в таблице

Типы команд SQL

- **DDL** (Data Definition Language – язык определения данных) создание и уничтожение объектов БД, обеспечение целостности данных (операторы CREATE TABLE/VIEW/INDEX, DROP/ TABLE/VIEW/INDEX, ALTER TABLE/INDEX).

CREATE DATABASE - создать базу данных
DROP DATABASE - удалить базы данных
CREATE TABLE - создать таблицу
ALTER TABLE - изменить таблицу
DROP TABLE - удалить таблицу
CREATE DOMAIN - создать домен
ALTER DOMAIN - изменить домен
DROP DOMAIN - удалить домен
CREATE VIEW - создать представление
DROP VIEW - удалить представление

Типы команд SQL

- **DCL** (Data Control Language – язык управления данными) – управление доступом пользователей к БД, а также назначение пользователям уровней привилегий доступа (операторы GRANT, REVOKE)

GRANT - предоставить привилегии пользователю или приложению на манипулирование объектами

REVOKE - отменить привилегии пользователя или приложения

Типы команд SQL

- **TPL** (Transaction Processing Language – язык обработки транзакций) – операторы COMMIT, ROLLBACK, SAVEPOINT

команды администрирования данных – аудит и анализ операций внутри БД, а также анализ производительности системы данных в целом (операторы START AUDIT, STOP AUDIT)

COMMIT - завершить транзакцию и зафиксировать все изменения в БД

ROLLBACK - отменить транзакцию и отменить все изменения в БД

SET TRANSACTION - установить некоторые условия выполнения транзакции

Создание запросов в СУБД Access средствами SQL

Запрос — объект базы данных, используемый для выборки или модификации хранимых данных.

В режиме конструктора можно открывать различные запросы: запрос на выборку, перекрестный запрос и запрос на изменение.

Запрос на выборку и перекрестный запрос также можно открыть в режиме таблицы для просмотра результатов.

Запросы на выборку и их использование

Запрос на выборку является наиболее часто используемым типом запроса. Запросы этого типа выбирает данные из одной или нескольких таблиц и отображают их в виде таблицы, записи в которой можно обновлять (с некоторыми ограничениями). Запросы на выборку можно также использовать для группировки записей и вычисления сумм, средних значений, подсчета записей и нахождения других типов итоговых значений.

Для подготовки запросов используются:

- QBE (Query By Example) — язык запросов по образцам,
- SQL (Structured Query Language) — язык структурированных запросов.

В основу языка структурированных запросов входят 4 основных оператора:

- SELECT – используется для выборки записей из таблиц;
- INSERT –используется для добавления записей в таблицу;
- UPDATE – используется для обновления записей таблицы;
- DELETE – используется для удаления записей из таблицы.

Оператор SELECT

Основой SQL является инструкция SELECT, используемая для создания запросов на выборку.

Синтаксис инструкции:

SELECT [ALL | DISTINCT |DISTINCTROW]
список_выбора

FROM имена таблиц

[WHERE критерий поиска]

[GROUP BY имя столбца, имя столбца,…]

[HAVING условие поиска]

[ORDER BY критерий столбца [ASC|DESC]];

SELECT — выбрать (директива) данные из указанных столбцов и (если необходимо) выполнить перед выводом их преобразование в соответствии с указанными выражениями и (или) функциями

FROM — из (условие) перечисленных таблиц, в которых расположены эти столбцы

WHERE — где (условие) строки из указанных таблиц должны удовлетворять указанному перечню условий отбора строк

GROUP BY — группируя по (условие) указанному перечню столбцов с тем, чтобы получить для каждой группы единственное агрегированное значение, используя во фразе SELECT SQL – функции: SUM (сумма), COUNT (количество), MIN (минимум), MAX (максимум), AVG (среднее значение)

HAVING — имея в результате лишь те группы, которые удовлетворяют указанному перечню условий отбора групп (условие)

ORDER BY — спецификация сортировки (условие) определяет порядок сортировки: ASC – сортировка по возрастанию, DESC - сортировка по убыванию.

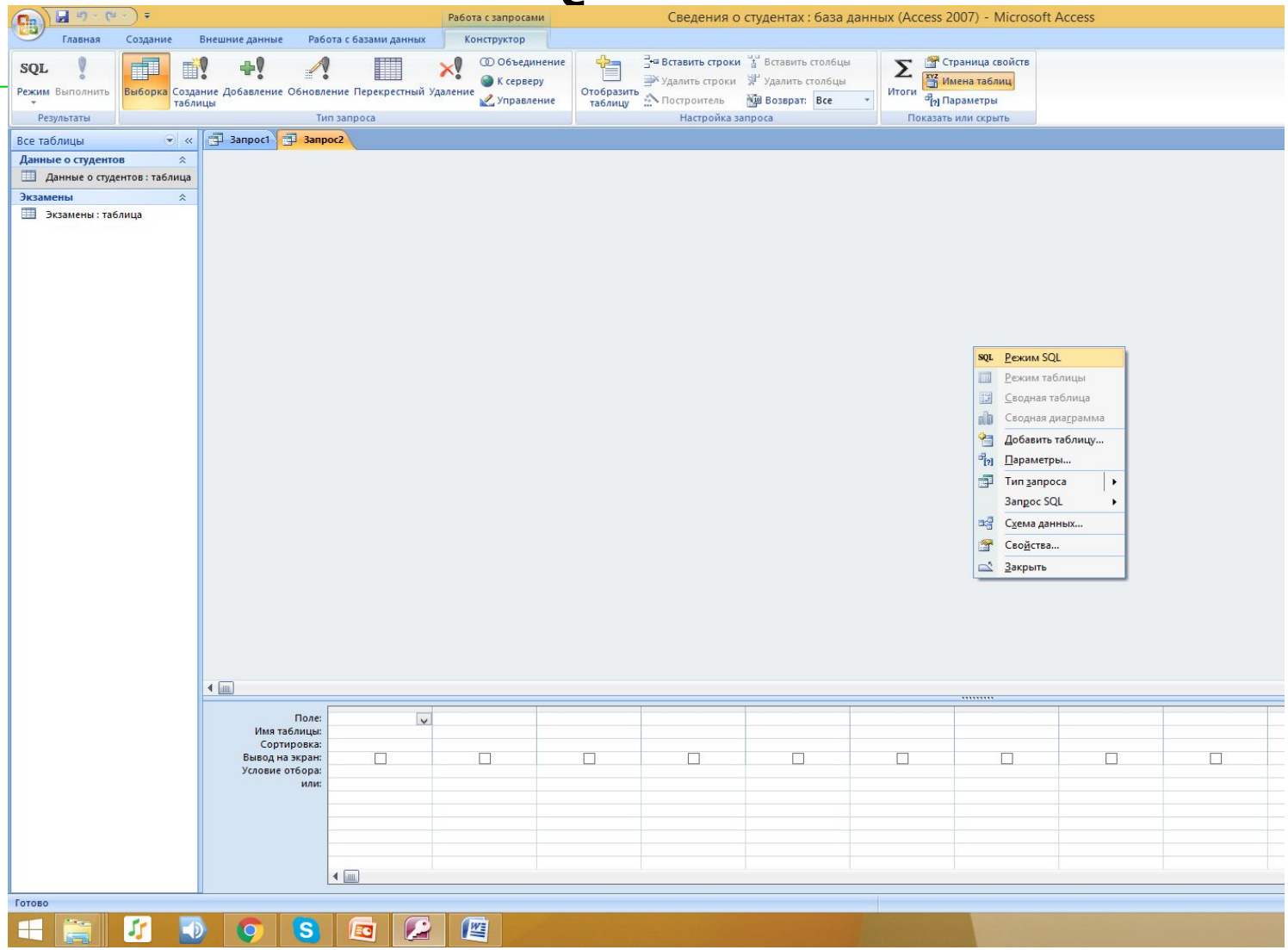
Запросы с использованием единственной таблицы

Все запросы на получение практически любого количества данных из одной или нескольких таблиц выполняются с помощью единственного предложения SELECT. В общем случае результатом реализации предложения SELECT является другая таблица. К этой новой (рабочей) таблице может быть снова применена операция SELECT и т.д., т.е. такие операции могут быть вложены друг в друга. Представляет исторический интерес тот факт, что именно возможность включения одного предложения SELECT внутрь другого послужила мотивировкой использования прилагательного "структуризированный" в названии языка SQL.

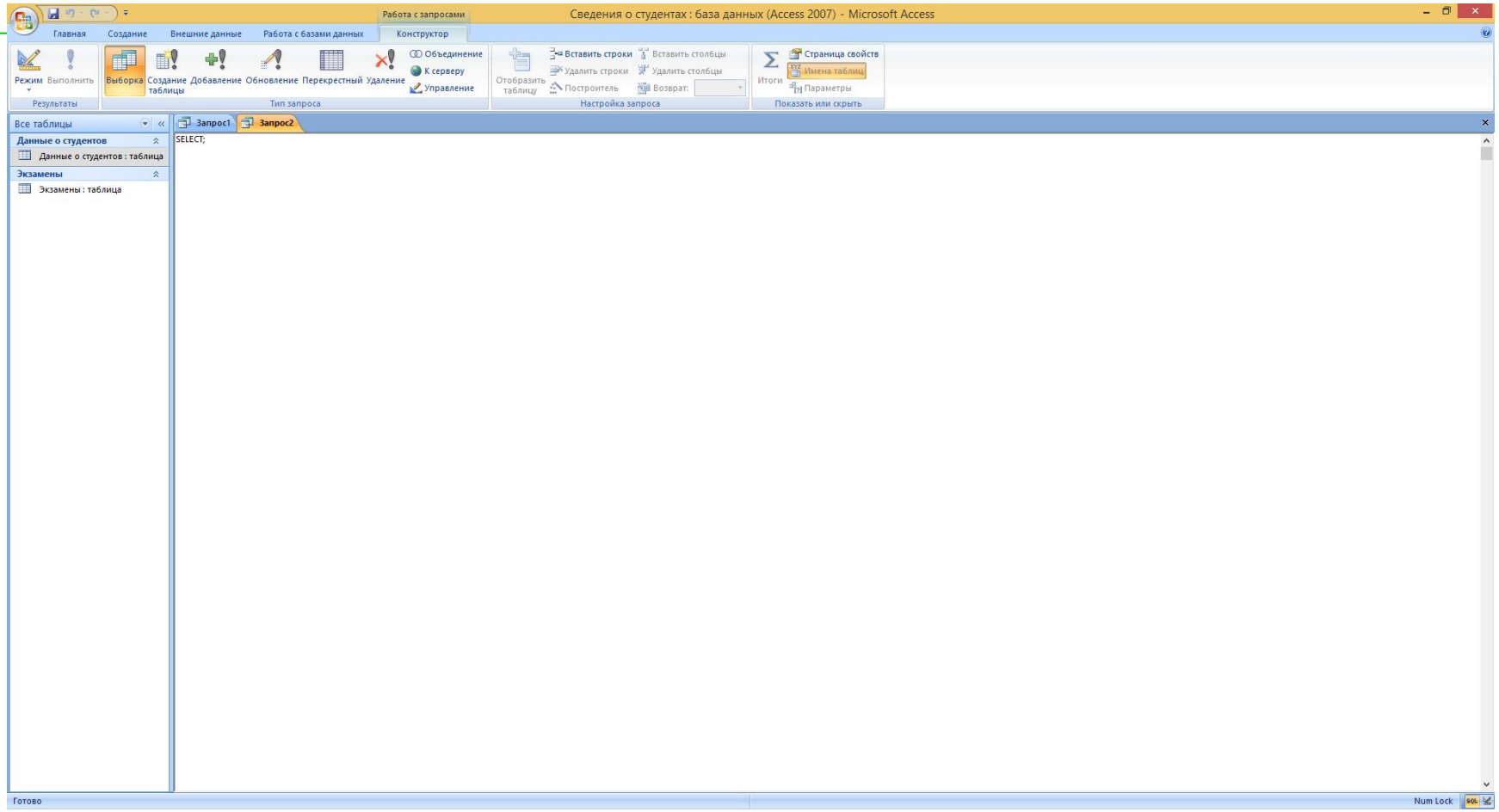
ПРЕДИКАТЫ

1. Сравнения =, <>, >=, <, <=
2. В интервале - “между” BETWEEN a1 and a2
3. Входит в множество IN (=[Предмет] IN (“История”,“Информатика”)
4. Подобие < имя > Like < образец >
   (что) (с чем сравнивать)

Режим SQL в MS Access



Окно SQL


Работа с запросами
Сведения о студентах : база данных (Access 2007) - Microsoft Access
Главная
Создание
Внешние данные
Работа с базами данных
Конструктор
Режим
Выполнить
Выборка
Создание таблицы
Добавление
Обновление
Перекрестный
Удаление
Объединение
К серверу
Управление
Отобразить таблицу
Вставить строки
Удалить строки
Построитель
Вставить столбцы
Удалить столбцы
Возврат:
Итоги
Страница свойств
Имена таблиц
Параметры
Результаты
Тип запроса
Настройка запроса
Показать или скрыть
Все таблицы
Данные о студентов
Данные о студентов : таблица
Экзамены
Экзамены : таблица
Запрос1
Запрос2
SELECT;
Готово
Num Lock

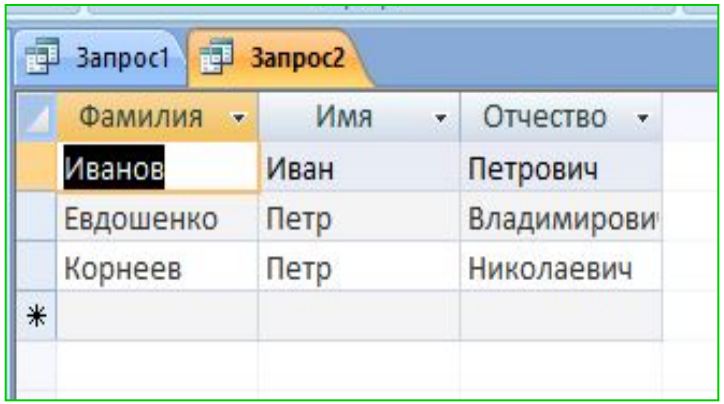
Рассмотрим синтаксис запросов на выборку:

1. Запрос на выборку фамилии, имени и даты рождения студента

   SELECT Фамилия, Имя, Отчество

   FROM Данные;

Результат:



При необходимости получения полной информации о Студенте, можно было бы дать запрос

SELECT Фамилия, Имя, Отчество, Город, Адрес, Телефон (и т.д.)

FROM Данные

или использовать его более короткую нотацию:

SELECT * (Звездочка (*) может применяться для вывода полного списка столбцов)

FROM Данные

Результат:

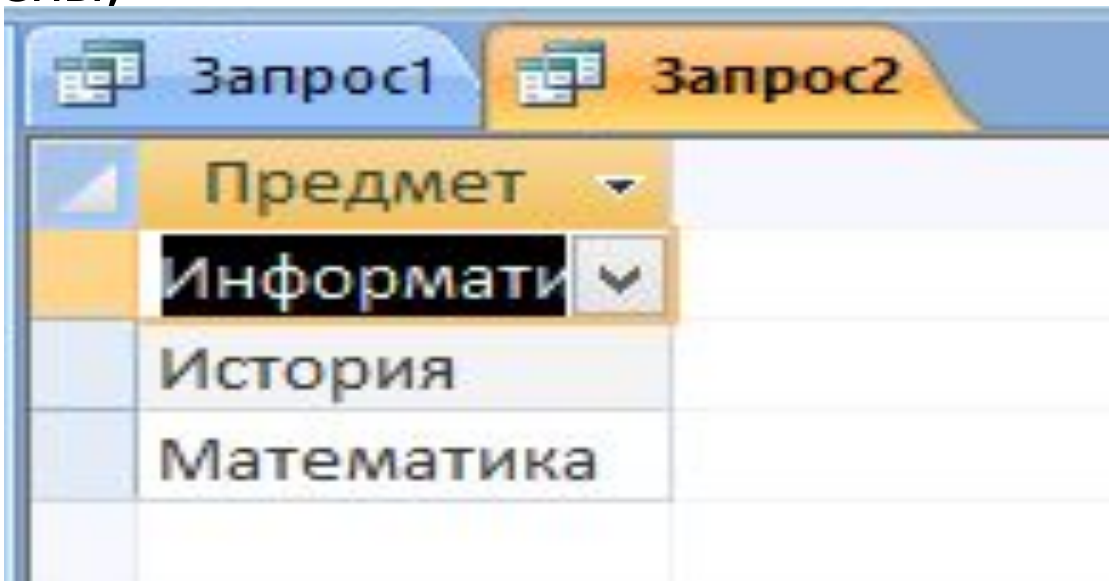
Запрос1

Номер заче	Фамилия	Имя	Отчество	Факультет	Курс	Группа	Дата_рожд	Стипендия	Город	Адрес	Телефон
2131	Иванов	Иван	Петрович	ФЭУ	2	14ЭУП	01.02.1997	2 500,00 р.	Оренбург	Загородное, 5	231254
2133	Евдошенко	Петр	Владимирови	ФЭУ	2	14ГиМУ	02.03.1998	1 980,00 р.	Гай	Полевая	215740
2405	Корнеев	Петр	Николаевич	ФЭФ	3	13ПИ	11.12.1996	4 500,00 р.	Оренбург	Победа	124441

Для исключения дубликатов и одновременного упорядочения перечня необходимо дополнить запрос ключевым словом DISTINCT (различный, различные), как показано в следующем примере:

```
SELECT DISTINCT Предмет;
FROM Экзамены;
```

Результат:

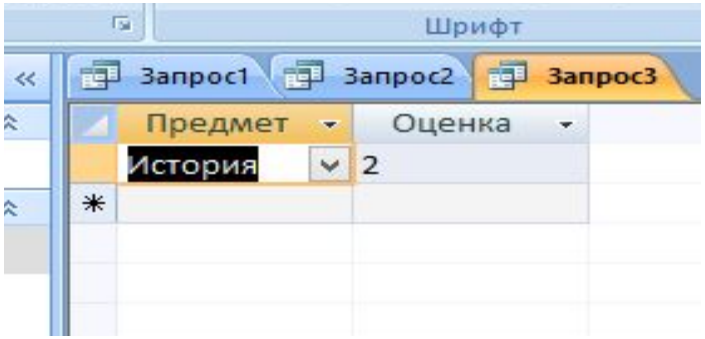


В синтаксисе фразы WHERE показано, что для отбора нужных строк таблицы можно использовать операторы сравнения = (равно), <> (не равно), < (меньше), <= (меньше или равно), > (больше), >= (больше или равно), которые могут предваряться оператором NOT, создавая, например, отношения "не меньше" и "не больше".

Так, для получения перечня предметов, по которым были получены 2, можно сформировать запрос

```
SELECT Экзамены.Предмет, Экзамены.Оценка
FROM Экзамены
WHERE (((Экзамены.Оценка)="2"));
```

Результат:



Оператор INSERT

```
INSERT INTO <имя_таблицы> [(<имя столбца_1> [,
  <имя столбца_1> ...])] {VALUES (<значение_1> [,
  <значение_2> ...]) | <выражение SELECT>};
```

Так, например, чтобы ввести строку в таблицу Продавцов, вы можете использовать следующее условие:

1. `INSERT INTO Salespeople VALUES (1001, 'Peel', 'London', .12);`
2. `INSERT INTO Customers (city, cname, cnum) VALUES ('London', 'Honman', 2001);`

Оператор UPDATE

Теперь, вы должны узнать как изменять некоторые или все значения в существующей строке. Это выполняется командой UPDATE.

```
UPDATE TABLE <имя_таблицы>
   SET <имя столбца_1> = <значение_1> [, <имя
   столбца_2> = <значение_2> ...]
   [WHERE <условие>];
```

Например

1. `UPDATE Customers SET rating = 200;`
2. `UPDATE Customers  SET rating = 200 WHERE snum = 1001;`
3. `UPDATE Salespeople SET sname = 'Gibson',city = 'Boston',comm = .10 WHERE snum = 1004;`

Оператор DELETE

Вы можете удалять строки из таблицы командой модификации - DELETE. Она может удалять только введенные строки, а не индивидуальные значения полей.

```
DELETE FROM <имя_таблицы> [WHERE <условие>];
```

Например

1. `DELETE FROM Salespeople WHERE snum = 1003;`
2. `DELETE FROM Salespeople WHERE city = 'London';`

Диалекты языка SQL в СУБД

- PL/SQL. Используется в Oracle. PL/SQL – это сокращение от Procedural Language/SQL. Он во многом похож на язык Ada.
- Transact-SQL. Используется в Microsoft SQL Server и Sybase Adaptive Server. По мере того как Microsoft и Sybase все больше отходят от общей платформы, которую они использовали в начале 90-х годов, их реализации Transact-SQL также подвергаются дивергенции.

Некоторые популярные диалекты SQL:

- **PL/pgSQL.** Название диалекта и расширений SQL, реализованных в PostgreSQL. Является сокра щением от Procedural Language/postgreSQL.
- **SQLPL.** Самый новый диалект от DB2 (SQLProcedural Language). Основан на стандартных операторах управления SQL. Большинство других диалектов предшествовало стандарту, и это означает, что вы найдете в них массу отличий от стандарта SQL.

Управление транзакциями

- Транзакция – последовательность логически связанных запросов, целенаправленно и логически связанно меняющих состояние БД;
- У транзакции имеется начало, набор точек сохранения (отката) и конец.
- В конце транзакцию можно применить (зафиксировать) или откатить
- В процессе исполнения транзакции, до ее завершения (фиксации или отката) объекты, которыми на манипулирует, могут быть «захвачены»

Операторы управления транзакциями:

- BEGIN – применяется для того, чтобы:
  - Зафиксировать, что транзакция началась
  - Указать (при необходимости), какие объекты захватываются и уровень их блокировки
- SAVEPOINT <NAME>
  - Указывает точку возврата, к которой можно откатиться при частичном откате транзакции
- RELEASE SAVEPOINT <NAME>

Операторы управления транзакциями:

- COMMIT – применяется для того, чтобы:
  - сделать «постоянными» все изменения, сделанные в текущей транзакции (реально данные могут быть изменены несколько позже)
  - очистить все точки сохранения данной транзакции
  - завершить транзакцию
  - освободить все блокировки данной транзакции

Операторы управления транзакциями:

- ROLLBACK – применяется для того, чтобы:
  - отменить все изменения, внесённые начиная с момента начала транзакции или с какой-то точки сохранения (SAVEPOINT).
  - очистить все точки сохранения данной транзакции
  - завершить транзакцию
  - освободить все блокировки данной транзакции

Примеры:

```
BEGIN;
  INSERT INTO table1 VALUES (1);
  SAVEPOINT my_savepoint;
  INSERT INTO table1 VALUES (2);
  ROLLBACK TO SAVEPOINT my_savepoint;
  INSERT INTO table1 VALUES (3);
COMMIT;
```

```
BEGIN;
  INSERT INTO table1 VALUES (3);
  SAVEPOINT my_savepoint;
  INSERT INTO table1 VALUES (4);
  RELEASE SAVEPOINT my_savepoint;
COMMIT;
```

Встраивание SQL в прикладные программы

Язык *SQL* можно использовать как в *интерактивном режиме*, так и путем внедрения его *операторов* в программы, написанные на процедурных языках высокого уровня.

Цель встроенных SQL – объединение возможностей языка программирования с реализованными посредством SQL механизмами работы со структурированными данными в БД.

Применение же языка *SQL* в прикладных программах на практике реализовано двумя различными способами:

Внедренные SQL-операторы. Отдельные SQL-операторы внедряются прямо в исходный текст программы и смешиваются с *операторами* базового языка. Этот подход позволяет создавать программы, обращающиеся непосредственно к базе данных. Специальные программы-предкомпиляторы преобразуют исходный текст с целью замены SQL-операторов соответствующими вызовами подпрограмм СУБД, затем он компилируется и собирается обычным способом.

Использование *прикладного интерфейса программирования* (API). Альтернативный вариант состоит в предоставлении программисту стандартного набора функций, к которым можно обращаться из создаваемых им программ. Конкретный вариант API может предоставлять тот же набор функциональных возможностей, который существует при подключении встроенных *операторов*, однако при этом устраняется необходимость предкомпилирования исходного текста.

Оба способа предполагают использование *операторов* как *статического SQL*, так и *динамического SQL*.

Операторы статического SQL - какого-либо изменения после их однократного написания не предполагается. Они могут храниться как в файлах, предназначенных для дальнейшего использования, так и в виде хранимых процедур *базы данных*.

Динамический *SQL* дает возможность программисту или конечному пользователю создавать *операторы* во *время выполнения* приложения и передавать их базе данных, которая после выполнения этих *операторов* помещает *выходные данные* в переменные программы.

Динамический SQL часто используется инструментальными средствами, предназначенными для построения заранее незапланированных запросов, позволяющих оперативно формировать тот или иной *оператор SQL* в зависимости от особых требований, возникших в конкретной ситуации. После настройки *оператора SQL* в соответствии с потребностями пользователя он направляется серверу баз данных для проверки на наличие синтаксических ошибок и необходимых для его выполнения привилегий, после чего происходит его *компиляция* и выполнение.

Команды SQL помещают в исходный код главной программы; - им предшествует фраза EXECSQL (Execute SQL); - далее устанавливаются некоторые команды, которые являются специальными для вложенной формы SQL.
Для вставки команды SQL в текст, написанный на другом языке, перед окончательной компиляцией необходимо выполнить прекомпиляцию.
Программы, называющиеся прекомпиляторами (препроцессорами), просматривают текст программы с преобразованием в форму SQL, удобную для исполнения базовым языком.
Обычно транслятор преобразует программу из исходно текста в исполняемый код.
Для пересылки данных из БД в программу используются спец. Команды SQL: Declare, Open, Fetch, Close, предназначенные для работы с курсором – переменной, связанной с запросом.
Declare – описывает выполняемый запрос и связывает имя курсора с результатом запроса. Определяет набор записей, в который будут возвращены результаты запроса.
Open – дает команду СУБД начать выполнение запроса и создавать таблицу результатов запроса.
Fetch – считывает данные запроса в переменную прикладной программы (считывает курсор из результатов запроса).
Close – прекращает доступ к таблице результатов запроса и ликвидирует связь между курсором и этой таблицей.

Администрирование БД

- По мере того как деятельность организаций всё больше зависит от компьютерных информационных технологий, проблемы защиты баз данных становятся всё более актуальными. Угрозы потери конфиденциальной информации стали обычным явлением в современном компьютерном мире. Если в системе защите есть недостатки, то данным может быть нанесен ущерб, который может быть выражен в: нарушении целостности данных, потере важной информации, попадании важных данных посторонним лицам и т.д.

Администратор - лицо, ответственное за целостность и непротиворечивость данных в системе, безопасность системы, эффективность функционирования системы и использования ею ресурсов. СУБД (система управления баз данных) "видит" администратора как пользователя, обладающего определенным набором привилегий. Привилегии администратора дают ему возможность использовать такие команды и утилиты СУБД и иметь доступ к таким системным таблицам, которые недоступны рядовым пользователям. Как правило, СУБД предоставляют в распоряжение администратора еще и специальный инструментарий, который обеспечивает удобный интерфейс для выполнения функций администратора.

функции администратора:

- инсталляция СУБД;
- управление памятью;
- управление разделением данных между пользователями;
- копирование и восстановление БД;
- управление безопасностью в системе;
- передача данных между СУБД и другими системами;
- управление производительностью.

Защита данных с использованием паролей

- Чтобы обеспечить защиту данных в компьютерных системах необходимо определить перечень мер, обеспечивающих защиту. Основными методами защиты баз данных являются защита ***паролем, шифрование, разграничение прав доступа***
- Защита паролем представляет собой простой и эффективный способ защиты БД от несанкционированного доступа. Пароли устанавливаются пользователями или администраторами БД. Учет и хранение паролей выполняется самой СУБД. Обычно, пароли хранятся в определенных системных файлах СУБД в зашифрованном виде. После ввода пароля пользователю СУБД предоставляются все возможности по работе с БД.

Защита данных с использованием шифрования

- Более мощным средством защиты данных от просмотра является их шифрование. ***Шифрование*** – это преобразование читаемого текста в нечитаемый текст, при помощи некоторого алгоритма; применяется для защиты уязвимых данных.

Разграничение прав доступа

- В целях контроля использования основных ресурсов СУБД во многих системах имеются средства *установления прав доступа* к объектам БД. Права доступа определяют возможные действия над объектами. Владелец объекта (пользователь, создавший объект), а также администратор БД имеют все права. Остальные пользователи к разным объектам могут иметь различные уровни доступа. Разрешение на доступ к конкретным объектам базы данных сохраняется в файле рабочей группы.

Восстановление БД

- Поскольку данные, хранимые компьютерными средствами подвержены потерям и повреждениям, вызываемым разными событиями, важно обеспечить средства восстановления данных. Приведение базы данных точно в то состояние, которое существовало перед отказом не всегда возможно, но процедуры восстановления базы данных могут привести ее в состояние, существовавшее незадолго до отказа.

- Для обеспечения защиты БД, используют различные ограничения целостности данных, журнализацию, репликацию и резервное копирование.

1. Резервное копирование базы данных

С целью обеспечения достоверности и постоянной работоспособности БД периодически вручную или автоматически осуществляется копирование базы данных. Основными причинами, побуждающими выполнение процедур копирования, являются различные структурные изменения базы данных:

- создание или удаление табличного пространства;
- добавление или переименование (перемещение) файла данных в существующем табличном пространстве;
- добавление, переименование (перемещение) или удаление журнала повторения и др.

2. Репликация

Репликацией БД называют создание специальных копий общей БД – *реплик*, с которыми пользователи могут одновременно работать на разных рабочих станциях. Пользователи могут вносить изменения в реплики, а потом обновлять общую БД, внося в неё изменения сделанные в репликах, то есть синхронизировать общую БД.

3. Журнализация изменений –

Это функция СУБД, которая сохраняет информацию, необходимую для восстановления базы данных в предыдущее состояние в случае логических или физических отказов.

Транзакция — это неделимая с точки зрения воздействия на БД последовательность операторов манипулирования данными (чтения, удаления, вставки, модификации).

В простейшем случае журнализация изменений заключается в последовательной записи во внешнюю память всех изменений, выполняемых в базе данных. Записывается следующая информация: порядковый номер, тип и время изменения; вид транзакции; объект, подвергшийся изменению (номер хранимого файла и номер блока данных в нём, номер строки внутри блока); предыдущее состояние объекта и новое состояние объекта.

Формируемая таким образом информация называется журнал изменений базы данных.

Характеристики серверов баз данных.

Современные серверные СУБД:

существуют в нескольких версиях для различных платформ, как правило, для различных коммерческих версий UNIX – Solaris, HP/UX. Многие производители также выпускают версии своих серверов баз данных для Windows NT Workstation Windows 95/98, а также версии для Linux;

в большинстве случаев поставляются с удобными административными утилитами;

осуществляют резервное копирование и архивацию данных и журналов транзакций;

поддерживают несколько сценариев репликаций;

позволяют осуществлять параллельную обработку данных в многопроцессорных системах. Серверы, допускающие параллельную обработку, разрешают нескольким процессорам обращаться к одной БД, что обеспечивает высокую скорость обработки транзакций;

поддерживают создание хранилищ данных и OLAP. Хранилище данных – это совокупность данных, полученных прямо или косвенно их информационных систем, которые содержат текущую и деловую информацию, а также из некоторых внешних источников.

выполняют распределенные запросы и транзакции;

дают возможность использовать различные средства проектирования схем данных – универсальные или ориентированные на конкретную СУБД;

имеют средства разработки клиентских приложений и генераторы отчетов;

поддерживают публикацию баз данных в Интернет;

обладают широкими возможностями управления пользовательскими привилегиями и правами доступа к различным объектам БД.

Механизмы доступа к данным базы на сервере.
Все серверные СУБД имеют клиентскую часть, которая обращается к БД посредством СУБД. Между клиентским прилож. и СУБД не существует прямой связи и дополнительно встраиваются программные модули, позволяющие клиентскому приложению получать доступ к БД, создаваемым с помощью разных СУБД. Такие модули называются механизмами доступа к данным. Существует 2 основных способа доступа к данным из клиентских приложений: использование прикладного интерфейса и использование универсального программного интерфейса. Универсальный механизм доступа к данным обеспечивает возможность использования одного и того же интерфейса для доступа к разным типам СУБД. Обычно он реализован в виде специальных дополнительных модулей, называемых драйверами. Наиболее распространенным программным интерфейсом, обеспечивающим доступ к данным конкретной базы данных является ODBC фирмы Microsoft. Для доступа к данной конкретн. СУБД, кроме клиентск. части необходимо приложение ODBC и драйвер.

Прикладной программный интерфейс (API) предст. соб. набор функций, вызываемых из клиентского приложения. Он может работать только с СУБД данного производителя и при ее замене придется переписывать значительную часть кода клиентского приложения.

Универсальный механизм доступа к данным обеспечивает возможность использования одного и того же интерфейса для доступа к разным типам СУБД. Обычно он реализован в виде специальных дополнительных модулей, называемых драйверами. Наиболее распространенным программным интерфейсом, обеспечивающим доступ к данным конкретной базы данных является ODBC фирмы Microsoft. Для доступа к данной конкретн. СУБД, кроме клиентск. части необходимо приложение ODBC и драйвер.

ODBC – открытый стандарт совместимости БД, разработанный в 1990-х для предоставления независимого от СУБД способа обработки информации из реляц БД. ODBC – интерфейс, с помощью которого прикладн проги могу обращ-ся к БД и обраб-ть независ от СУБД способом. ODBC-драйвер выполняет все вызовы ODBC-функций и «переводит» их на язык источника данных. СУБД хранит и выводит данные в ответ на запросы со стороны ODBC-драйвера. Приложение ODBC – для определения доступа источника данных для конкрет. компа и описания источн. данных. Задание ODBC-источника данных является действием, которое осуществляется средствами операционной системы, управляющей компьютером. Источник данных – структура дан. ODBC, идентифицирующая БД и СУБД, которая ее обрабатывает.

С его помощью могут быть заданы: *1.пользовательский* – ист. дан., доступный только текущему пользователю на текущем компьютере; *2.файловый* – фал, кот. может совместно использоваться пользователем БД; *3.системный* – источник данных, доступный всем пользователям и службам текущего компьютера.
Преимущества: - простота разработки приложения; - позволяет создавать распределенные гетерогенные приложения без учета конкретной СУБД – приложения становятся независимыми от СУБД.
Недостатки: - снижение скорости доступа к данным; - увеличение времени обработки запросов; - предварит инсталляция и настройка ОДВС на кждом рабочем месте; - представляет доступ только к SQL-ориентированным БД.
OLEDB и ADO – осн. часть универс-го мех. доступа к данным фирмы Майкрософт, позволяющ. осуществить доступ к реляцион. и нереляцион. источн. данных. OLEDB – реализация разработанного Майкрософт объективного стандарта OLE. Для доступа к источнику дан. С пом. OLEDB треб-ся на клиентском компе установаить провайдер для данной СУБД. Механизм доступа к данным ADO: – высокоуровнев. программн. интерфейс для доступа к дан. из приложения; - Содержит набор объектов, исп-емых для соединения с источником дан., чтения, добавления и модификации данных.

Спасибо за внимание